SAND99-0466
Unlimited Release
Printed March 1999

MAPVAR - A Computer Program to Transfer Solution Data Between Finite Element Meshes

Gerald W. Wellman
Engineering and Manufacturing Mechanics Department
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-0443

Abstract

MAPVAR, as was the case with its precursor programs, MERLIN [1] and MERLIN II [2], is designed to transfer solution results from one finite element mesh to another. MAPVAR draws heavily from the structure and coding of MERLIN II, but it employs a new finite element data base, EXODUS II [3], and offers enhanced speed and new capabilities not available in MERLIN II. In keeping with the MERLIN II documentation, the computational algorithms used in MAPVAR are described. User instructions are presented. Example problems are included to demonstrate the operation of the code and the effects of various input options.

.

Acknowledgment

The author wishes to thank D. Gartling, the author of the MERLIN and MERLIN II codes, for generously providing source code, ideas, and consultations. In addition the author would like to thank M. Heinstein for providing the JAS3D contact surface binary search algorithm. Finally the author would like to thank those users who expressed their needs for data transfer and performed much of the code testing and verification, in particular B. Thorne, R. Chambers, M. Heinstein, and R. Shunk.

Contents

List of Figures

List of Tables

This Page Intentionally Blank

Introduction

Transfer of finite element results from one mesh to another is typically utilized for two classes of problems. In the solution of weakly coupled multi-physics problems, the results from one simulation would be used as input for the next simulation. For example, the temperature field from a thermal simulation would be used as the input to compute thermal stresses in a structural simulation. In weakly coupled multi-physics problems, the different phenomena to be simulated typically require different degrees of mesh resolution and thus the transfer of results between the meshes. The use of rezoning/adaptivity in solution schemes also requires the transfer of results from one mesh to another. Rezoning is typically identified as a method to correct unacceptable distortion in a Lagrangian mesh. For example, large deformations can lead to distortion or even inversion of finite elements in the structural simulation of impact events or manufacturing processes. Adaptivity is generally described as the use of selective refinement of the mesh to capture evolving or translating phenomena for example strain localization or a moving burn front. MAPVAR is intended to be the data transfer module for both these applications.

MAPVAR draws heavily from its precursor programs, MERLIN [1] and MERLIN II [2]. The principal differences between MAPVAR and MERLIN II are shown in Table 1.

Table 1. Principal Differences Between MERLIN II and MAPVAR

	MERLIN II	MAPVAR
Database	EXODUS/GENESIS [4]	EXODUS II [3]
Element Types	Multiple	4- and 9-Node Quad, 4-Node Shell, 8-Node Hex
Search	Boxed	Binary
Interpolation	Nodal Averaging	Nodal Averaging, Constrained Least Squares, None
Variable Type	Nodal	Nodal, Element
Data Map	None	Element Block by Element Block
Geometry	Original	Original, Deformed, Annealed
Points Outside Region	Set to “Halo” Value	User Subroutine
Number of Variables	20 - Fixed Dimensions	No Limit - SUPES[5] Dynamic Memory Allocation

MAPVAR retains in common with MERLIN II the requirement that the two meshes occupy the same physical region. That is, both meshes have a common origin, the same units of length, and the same boundaries. Only <u>very</u> limited extrapolation is allowed. The ability to utilize a wide range of finite element types is retained in MAPVAR. However, only those few element types shown in Table 1 have been implemented to date. This limited set of element types reflects the needs of the current users. Budgetary constraints dictate that element type capability only be added as needed.

The intent of MAPVAR is to bring the functionality embodied in MERLIN II into line with the current state of SEACAS [6] primarily through the use of the dynamic memory allocation provided by SUPES and through the use of the finite element database manipulation capabilities of EXODUS II. At the same time, it was possible to significantly improve efficiency and speed of operation of MAPVAR over that of MERLIN II. Finally, additional functionality has been added based on the requirements of specific programs including Advanced Strategic Computing Initiative (ASCI) neutron generator certification, ASCI encapsulation, Advanced Computational Technology Initiative (ACTI), GOMA [9] and Goodyear tire wear. Complete generality of functionality has been sacrificed to constraints of available manpower and funding. However, the modular nature of MAPVAR makes it easy to add capability.

Search Algorithm

MERLIN II uses a classical bucket search algorithm. This type of search can be very effective but it suffers from a few drawbacks. A bucket search requires the buckets to be defined in some way. In MERLIN II, the user was required to define the number of buckets in each ordinal direction. A very knowledgeable user could often select a near optimal bucket spacing. However, a novice user could actually reduce, by injudicious definition of buckets, the efficiency of the code over not using any buckets at all. In addition, the bucket search is inefficient for any geometry which mandates that certain buckets contain many nodes while other buckets contain few or even no nodes.

The new search algorithm is a variation on a binary search proposed by Swegle [7]. The actual coding used in MAPVAR was derived from the contact search subroutines described by Heinstein, et. al. [8]. As implemented in MAPVAR, the search algorithm can be briefly described as:

1. Sort the points (nodes or element centroids) of the recipient mesh based on coordinates. Produce two sequential lists for each coordinate (dimension) in the recipient mesh. The first list (index list) provides the identifying number (node or element number) for the location in the sorted list. Thus the first location in the index list has the node or element number that has the lowest value of that coordinate. The second list (rank list) gives the array location for each node or element number in the index list. For example, if node number 5 has the lowest x-coordinate value in the recipient mesh then the index list has 5 as its first entry and the rank list has 1 as its fifth entry.

2. Compute the minimum and the maximum value for each coordinate for each element in the donor mesh. Increase the interval thus computed by a tolerance (the searchbox). Use a binary search on the index list to provide the integer value that defines the first and last entries that lie within the searchbox.

3. Do an intersection of the list segments for each coordinate. This provides a list of recipient mesh points that could lie within the donor mesh element.

4. Compute the isoparametric coordinates within the donor mesh for each of the recipient mesh points.

5. After looping over steps 3, 4, and 5 for all donor mesh elements, pick the donor mesh element with the best set of isoparametric points and save those values for future interpolation of the data.

Readers who desire a more complete description of the search are directed to the above references.

This Page Intentionally Blank

User Input

Help

The **HEL**p command provides the list of MAPVAR commands to the standard output device, typically the terminal screen. The list of commands is:

HELp

HELp <keyword>

SCHeme <int>

DEFormed <int>

LISt **TIM**es

TIMes <real or **ALL**>

SEArchbox <real>

MAP <int or **ALL**> **TO** <int or **ALL**> **SCH**eme <int>

MAPVAR commands take the form of a keyword followed by a value. The portion of the keyword or value shown in boldface above is the minimum number of characters required to be input. All character string parsing is done for the first three characters with the remainder of the string ignored. The item within the pointed brackets represents the input expected for any given keyword. A <int> refers to the need for an integer value to be entered after the keyword. Likewise, a <real> indicates the need for a real number input. Two commands, the **TIM**es and the **MAP** commands will accept either the character string, **ALL**, or a number (real or integer respectively) as input. The **HEL**p keyword command where keyword is any of the character string keywords listed above will provide a brief description of the command to the standard output device. A detailed description of the available MAPVAR commands is presented below.

Interpolation Schemes

The command line to control the interpolation scheme is:

SCHeme <int>

At present, MAPVAR retains the use of element shape functions to define the value of the variable to be assigned to the point from the recipient mesh which lies within an element of the donor mesh. This procedure requires the values of the variable to be defined at the nodes of the donor mesh. In the context of MAPVAR, interpolation scheme is defined as the method to transform element centroid (element variable) data to nodal data. Three interpolation schemes are currently implemented in MAPVAR. These are simple nodal averaging, constrained linear least squares, and direct transfer which does no interpolation. Each of these schemes has advantages and disadvantages which will be discussed in detail below. The “best” scheme is completely problem dependent and thus the choice of scheme is left up to the user.

SCHeme 0

Scheme 0 is a simple nodal averaging scheme. In this scheme, the value of an element variable at a node is determined by averaging the values of that variable from the centroid of each of the elements that share that node. The algorithm in MAPVAR progresses as follows:

1. Loop over all the elements in the element block being processed.

2. Loop over all the nodes for each element.

3. Accumulate the sum of each element variable at each node. Keep a counter of the number of contributions to the sum at each node.

4. Divide the sum by the number of contributions to obtain an average nodal value for each element variable.

This scheme has the advantage of being fast, robust, and simple. In the interior of a structure it provides a very accurate interpolation. However, this scheme will result in dissipation or smoothing of the variable values near the structure boundary for variables which exhibit gradients. In particular, when the recipient mesh has a finer grid than the donor mesh, this scheme is not recommended when the solution exhibits a gradient in element variables toward the boundary of an element block.

SCHeme 1

This is the default scheme. Scheme 1 scatters element centroidal variable values to the node through a linear constrained least squares fitting procedure. Scheme 1 starts by forming an inverse connectivity matrix - a list of all elements connected to a node. After some checks for potentially pathological cases, a linear least squares procedure is employed to compute the value of the variable at the nodal location given the location of and the variable value at the surrounding element centroids. Finally, the constraint phase replaces physically meaningless values of variables, such as negative plastic strain, with an appropriate default value. A summary of the Scheme 1 algorithm follows:

1. Create inverse connectivity.

2. If not enough elements connected to node (at least 3 for 2-D, 5 for 3-D), pick node on opposite side of element. If still not enough elements, do simple average for the original node (identical to Scheme 0).

3. If elements are colinear (2-D) or coplanar (3-D), again do simple averaging.

4. If enough elements have been located for this node and these elements are not coplanar, then do a linear least squares fit centered on this node using the location of and the value at the element centroids.

5. Go back and fix any values that are physically unacceptable.

Scheme 1 is the most costly of the interpolation schemes. However, run times for MAPVAR should rarely be a concern. Scheme 1 does some extrapolation near boundaries, particularly if the recipient mesh is more finely gridded than the donor mesh. This is both an advantage and a disadvantage. Extrapolation of element variable results is inherently dangerous. However, for the case of a more finely gridded recipient mesh, extrapolation is the only way to avoid significant dissipation/smoothing. The imposition of external constraints on the results of the least squares fit will mitigate some of the worst aspects of extrapolation. Scheme 1 is the best method currently implemented to capture gradients in a solution. Over the range of applications tried thus far, Scheme 1 generally provides the best interpolation of element variables and thus has been selected as the default scheme.

SCHeme 2

Scheme 2 is based on direct element to element variable transfer. Thus there is no intermediate step of interpolation of element variables to nodes. Instead, if the element centroid of the recipient mesh lies within a donor mesh element, the recipient mesh element is given the value of the donor mesh element. This procedure is the fastest of the interpolation schemes, no interpolation is performed. It has the advantage that for identical donor and recipient meshes, the solution for the donor mesh is preserved exactly. However, Scheme 2 is not recommended if the donor mesh differs from the recipient mesh, particularly if the recipient mesh is more finely gridded than the donor mesh. In this case, Scheme 2 will effectively sharpen gradients in element variables. That is, the difference in the variable value between two elements will be identical while the spacing between the two element centroids is decreased. Scheme 2 is recommended for identical donor and recipient meshes and can be used with care for a recipient mesh coarser than the donor mesh.

Geometric Processing

The command line to control the selection of deformed versus undeformed geometry is:

DEFormed <int>

There are three options for the deformed processing command. The user may select to do all processing in the undeformed geometry, the deformed geometry, or may select mesh annealing. These options are discussed in more detail below.

DEFormed 0

In this option, all coordinates used in MAPVAR are the original coordinates. Donor mesh displacements are not considered. The recipient mesh boundaries should coincide with the boundaries of the original finite element mesh that was used to produce the donor mesh. Displacements are considered to be just another nodal variable, nothing more. This option can be useful for small deformation problems to avoid the necessity of defining the deformed shape of the structure in a manner suitable for use in generating a new mesh. If undeformed processing is selected, then the structural shape used to produce the original mesh may be used to produce all subsequent meshes.

DEFormed 1

This is the default option. In this option, MAPVAR processing is performed in current, deformed coordinates. The boundaries of the recipient mesh are consistent with the deformed shape of the donor mesh. During processing, the displacements from the donor mesh are read and added onto the coordinates of the donor mesh to produce current coordinates. Just prior to writing the interpolated mesh, the transferred displacements are subtracted from the recipient mesh coordinates. This produces an interpolated mesh that, at the time step of interest, has current coordinates identical to the recipient mesh. That is, when plotted with a displacement magnification of one, the interpolated mesh is identical to the recipient mesh. This technique allows the displacement field to remain meaningful across a rezoned restart. This is particularly useful when mesh distortion requires rezoning in order for the solution to proceed. The interpolated mesh provides for a restart with well shaped elements in current coordinates while still retaining the meaning of the displacement field.

DEFormed 2

This option is called mesh annealing. It was supplied in support of the GOMA [9] code. In mesh annealing, the data transfer is performed in deformed, current coordinates. Thus, during processing, the donor mesh displacements are added to the donor mesh coordinates to produce the required current coordinates. The boundaries of the recipient mesh are coincident with the deformed boundaries of the donor mesh. However, prior to writing the interpolated solution, the displacements are zeroed. This produces an interpolated mesh that while consistent with the deformed shape of the donor mesh has a zero displacement field. For further discussion of the applications for this option, the reader is directed to [9].

At present, only a single recipient mesh may be supplied. Therefore, undeformed processing is the only option available when multiple time steps are to be transferred. There is simply no way to input a series of recipient meshes that track the deformation at each step of the solution.

Time Step

The command to search the donor mesh database and determine the available time steps is:

LISt **TIM**es

This command will write a list of the time step (integer number that varies from one to the number of time steps available) followed by the time associated with that time step (a real number that varies from the minimum time available - typically zero to the maximum time available).

The command for selecting the time step of interest is:

TIMes <real or **ALL**>

If a real number is input after the **TIM**es keyword, that real number is processed to determine the time step associated with the closest time available from the donor mesh data base. If the character string **ALL** is input, then all the time steps available from the donor mesh data base are selected. Only the option to select all the time steps or to select a single time step is currently available. If a set of time steps, different from all available time steps, is required, ALGEBRA[10] may be used to provide a donor mesh with only the desired time steps. The default for this command is the last time step available from the donor mesh.

Searchbox

The command to control the searchbox for the search routine is:

SEArchbox <real>

The searchbox is essentially a tolerance on the minimum and maximum value of each coordinate for each element or group of elements from the donor mesh sent into the search routine. The interval for each coordinate is computed from the maximum and minimum above. This interval is multiplied by the value entered via the searchbox command to obtain a local search tolerance. This tolerance is then subtracted from the minimum and added to the maximum values for use in the search routine. A searchbox value of less than zero is not recommended. Increasing the searchbox value increases the cost of the search. Thus far, the major use for an increased value of the searchbox is to deal with the problem of different intervals along a curved surface. Typically, when a curved surface is discretized, the nodes are constrained to lie on the surface. Thus for different discretizations, the nodes of one discretization will lie outside the straight line segments defined by the elements of the other discretization. Modest increases in the searchbox parameter can alleviate this problem.The default value for the searchbox is 0.01. Searchbox values greater then one are not recommended because this will significantly slow down the search routine.

MAP

Element and nodal variables are transferred from the donor mesh to the recipient mesh element block by element block. This type of transfer prevents blending of material properties and results across material boundaries. It also permits the correct treatment of displacements for surfaces that have come into contact. In EXODUS II each element block (typically each material occupies a unique element block) has a unique identifier. MAPVAR permits the user to define the correspondence among element blocks between the donor mesh and the recipient mesh. The user can also specify the interpolation scheme to be used for each element block transfer. The command to specify a user defined element block to element block transfer is:

MAP <int> **TO** <int> **SCH**eme <int>

Once a **MAP** command has been entered, only those element blocks identified through this and subsequent **MAP** commands will be transferred. That is, a **MAP** command does not update a predefined transfer sequence but instead increments the user defined transfer sequence. The **MAP** command is very general, both transfers from many element blocks to one and one element block to many are permitted. MAPVAR attempts to provide some useful information to the user of this option. Whenever a **MAP** command is entered, MAPVAR will echo a list of all the user entered element block transfers defined thus far. MAPVAR will also provide a list of the recipient mesh element blocks not currently identified for transfer. MAPVAR provides a default one to one transfer map. The default transfer definition is determined by looping over all the element blocks in the recipient mesh. The identification numbers for the recipient mesh element blocks are matched to the identification numbers of the element blocks in the donor mesh. If a match is found, those matching element blocks are defined in the default transfer map. For example, element block I. D. 1 in the donor mesh will be mapped into element block I. D. 1 in the recipient mesh. If no match is found, for element blocks in either recipient or donor meshes, the variables in those element blocks will not be transferred.

The interpolation scheme can also be set for each element block independently through the **MAP** command. Any of the three schemes described above can be selected. If a value other than 0, 1, or 2 is entered using the **MAP** command, the local scheme reverts to the current default value. The current default interpolation scheme is either the global default, 1, or whatever value the user last entered via the **SCH**eme command.

User Subroutine

A user subroutine is provided to allow the user to set the value of either nodal or element results for recipient mesh nodes or element centroids that lie outside the boundaries of the donor mesh or for whatever reason are not included in the transfer map. It was originally intended to call the user subroutine only for those points not found in the search. However, for the case where one donor mesh element block is mapped into many recipient mesh element blocks, at the level of the search routine many points will not be found for each search. Rather than increase the required storage, it was decided to initialize all points prior to commencing the search. This prior initialization has not been found to be excessively expensive. By default, all variables associated with a node or element centroid are initialized to zero.

The user subroutine for nodal values is ininod.f. The default subroutine is shown below:

```
      SUBROUTINE ININOD(SOLNB,IVAR,TIMES,ISTP,IDBLK,NDLSTB,XB,YB,ZB)
C
C ***********************************************************************
C
C ININOD initializes nodal variable values based on TIME, ELEMENT
C BLOCK, VARIABLE NAME, COORDINATE, etc. By default, nodal variable
C values are set to zero. It is intended that the user rewrite this
C subroutine to provide values that are appropriate to the problem
C being solved. This is the preferred method to handle nodal variable
C assignment for recipient mesh nodes that lie outside the boundary
C of the donor mesh.
C
C
C Called by INTRPN, SINTPN
C
C ***********************************************************************
C
C SOLNB   REAL Array of nodal variable values
C              (1:nodesb,1:nvarnp,1:ntimes)
C IVAR    INT   Position of variable in SOLNB array
C TIMES   REAL Array of times (1:ntimes)
C ISTP    INT   Position in TIMES array, time step being processed
C IDBLK   INT   The element block I. D.
C NDLSTB INT   Array of nodes that belong to this element block
C              (1:numndb)
C XB      REAL Array of X-coordinates (1:nodesb)
```

```
C YB      REAL Array of Y-coordinates (1:nodesb)
C ZB      REAL Array of Z-coordinates (1:nodesb)
C
C ******************************************************************
C
      include ‘exodusII.inc’
      CHARACTER*(MXSTLN) QALINE,NAMECO,NAMVAR,NAME
C
      COMMON /AEXDS1/ NQAREC,NVARGP,NVARNP,NVAREL
      COMMON /BMESH/ NUMELB,NODESB,NBLKSB,NDIMB,NELNDB
      COMMON /EBBYEB/ NUMEBA,NUMEBB,NUMNDA,NUMNDB,ITYPE
C
      DIMENSION SOLNB(NODESB,NVARNP), TIMES(*), NDLSTB(*)
      DIMENSION XB(*), YB(*), ZB(*)
C
C ******************************************************************
C
C Code to help you find some potentially useful stuff
C The actual time (real number)
C     TIME = TIMES(ISTP)
C
C The pointer into VARNAM to get the variable name being processed
C     INAM = IVAR + NVARGP + NVAREL
C
C The name of the variable (character) being processed
C     NAME = NAMVAR(INAM)
C
C INOD = NDLSTB(I)
C
C Set value of node-NDLSTB(I); variable-IVAR to 0. by default.
C User to replace this with whatever code he wishes.
C
      DO 10 I = 1, NUMNDB
        INODE = NDLSTB(I)
        SOLNB(INODE,IVAR) = 0.
 10   CONTINUE
C
      RETURN
      END
```

The user subroutine for element variables is very similar as shown below:

```
      SUBROUTINE INIELT(SOLEB,IVAR,TIMES,ISTP,IDBLK,CENTER)
C
C ******************************************************************
```

```
C
C INIELT initializes element variable values based on TIME, ELEMENT
C BLOCK, VARIABLE NAME, COORDINATE, etc. By default, element variable
C values are set to zero. It is intended that the user rewrite this
C subroutine to provide values that are appropriate to the problem
C being solved. This is the preferred method to handle element variable
C assignment for recipient mesh nodes that lie outside the boundary
C of the donor mesh.
C
C
C Called by INTRPE, SINTPE, TRANAB, STRAN
C
C *********************************************************************
C
C SOLEB   REAL Array of element variable values
C              (1:numebb,1:nvarel)
C TIMES   REAL Array of times (1:ntimes)
C IDBLK   INT   The element block I. D.
C CENTER REAL Array of element centroid coordinates
C              (1;numebb,1:3)
C
C *********************************************************************
C
      include 'exodusII.inc'
      CHARACTER*(MXSTLN) QALINE,NAMECO,NAMVAR,NAME
C
      COMMON /AEXDS1/ NQAREC,NVARGP,NVARNP,NVAREL
      COMMON /EBBYEB/ NUMEBA,NUMEBB,NUMNDA,NUMNDB,ITYPE
C
      DIMENSION SOLEB(NUMEBB,NVAREL), TIMES(*), CENTER(NUMEBB,*)
C
C *********************************************************************
C
C Code to help you find some potentially useful stuff
C The actual time (real number)
C     TIME = TIMES(ISTP)
C
C The pointer into VARNAM to get the variable name being processed
C     INAM = IVAR + NVARGP
C
C The name of the variable (character) being processed
C     NAME = NAMVAR(INAM)
C
C The coordinates of the point (element centroid)
C
C XP = CENTER(IELT,1)
```

```
C YP = CENTER(IELT,2)
C ZP = CENTER(IELT,3)
C
C By default, set value to 0.
C User to replace this with whatever code he wishes.
C
      DO 10 IELT = 1, NUMEBB
        SOLEB(IELT,IVAR) = 0.
   10 CONTINUE
C
      RETURN
      END
```

In both these subroutines the variable to be mapped, along with a way to determine its variable name, the time step and a way to determine the corresponding time, the element block identification number, and the coordinates of the point (either node or element centroid) are made available through the argument list and through common blocks. If additional information is required by the user, the author of this report or the code sponsor should be contacted.

Example Problems

The first example is a square bar tensile specimen. In order to generate gradients in stress and strain, the bottom of the bar was fixed in all directions, the top of the bar had a Y-direction displacement applied but was allowed to displace freely in the X and Z-directions. The initial undeformed mesh was a square consisting of 8 elements in a 2 by 2 by 2 element arrangement shown in Figure 1.

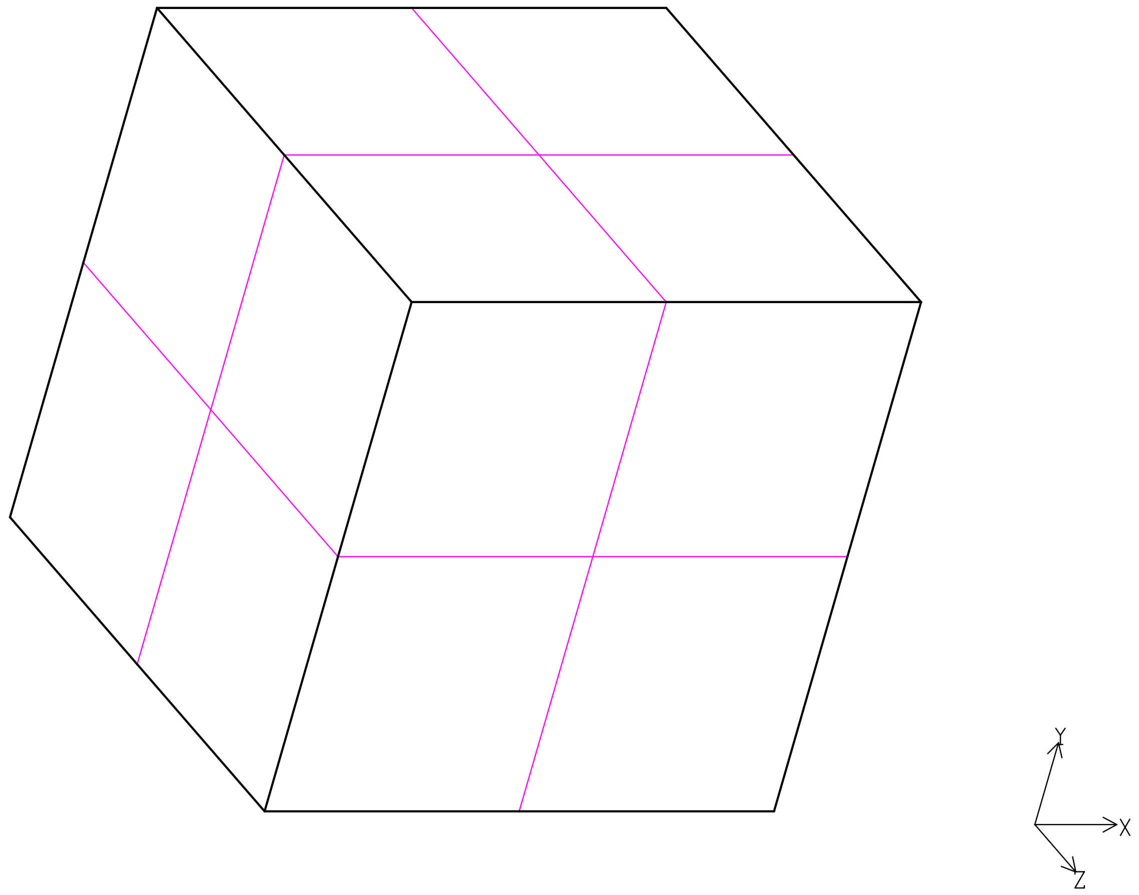

Figure 1. Initial, undeformed mesh for square tensile specimen

The bar was modeled as an elastic-plastic material with the applied displacement well beyond the elastic limit. Effective plastic strain (EQPS) was chosen as the element variable to be transferred to the new mesh. The “epaint” option in BLOT [12] was chosen to display the effective plastic strain. This option paints the entire element with the appropriate color for the value of the element variable at the centroid of the element. Figure 2 shows the effective plastic strain “epainted” on the deformed shape. The recipient mesh onto which the effective plastic strain is to be transferred is more finely gridded as shown in Figure 3.

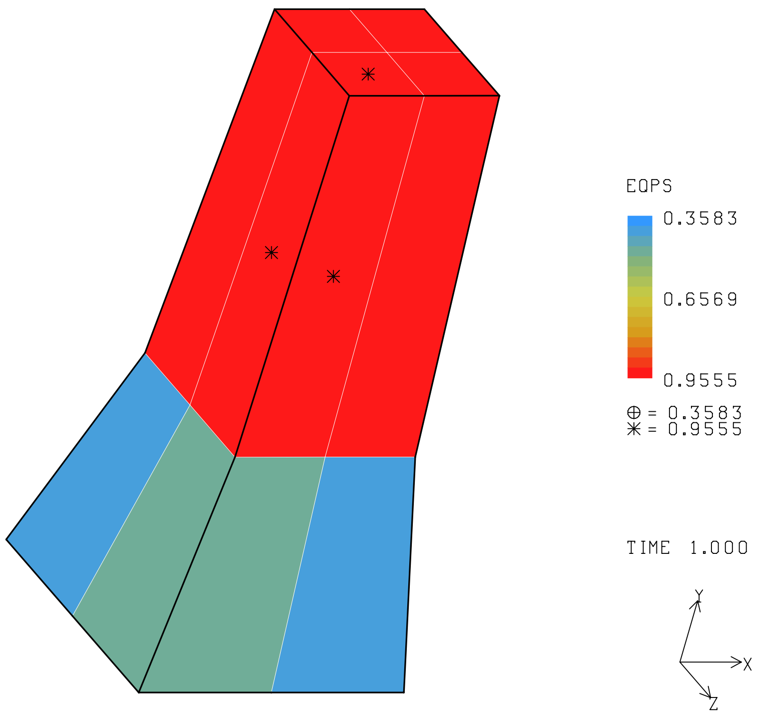

Figure 2. Effective plastic strain “epainted” on the deformed shape for the square tensile specimen.

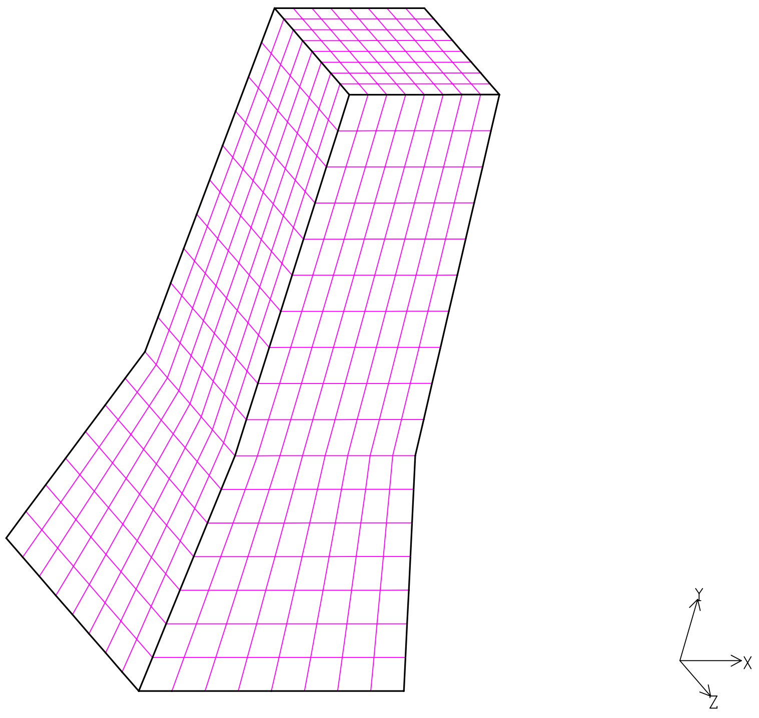

Figure 3. Recipient mesh for the square tensile bar.

The results of the data transfers are shown in Figure 4. The effective plastic strain on the original mesh (repeat of Figure 2) is shown in the upper left-hand corner. The result of using scheme 0 (simple interpolation) for the transfer is shown in the upper right-hand corner. The result of using scheme 1 (linear least squares interpolation) is shown in the lower left-hand corner. Finally, the result of using scheme 2 (direct transfer) is shown in the lower right-hand corner.

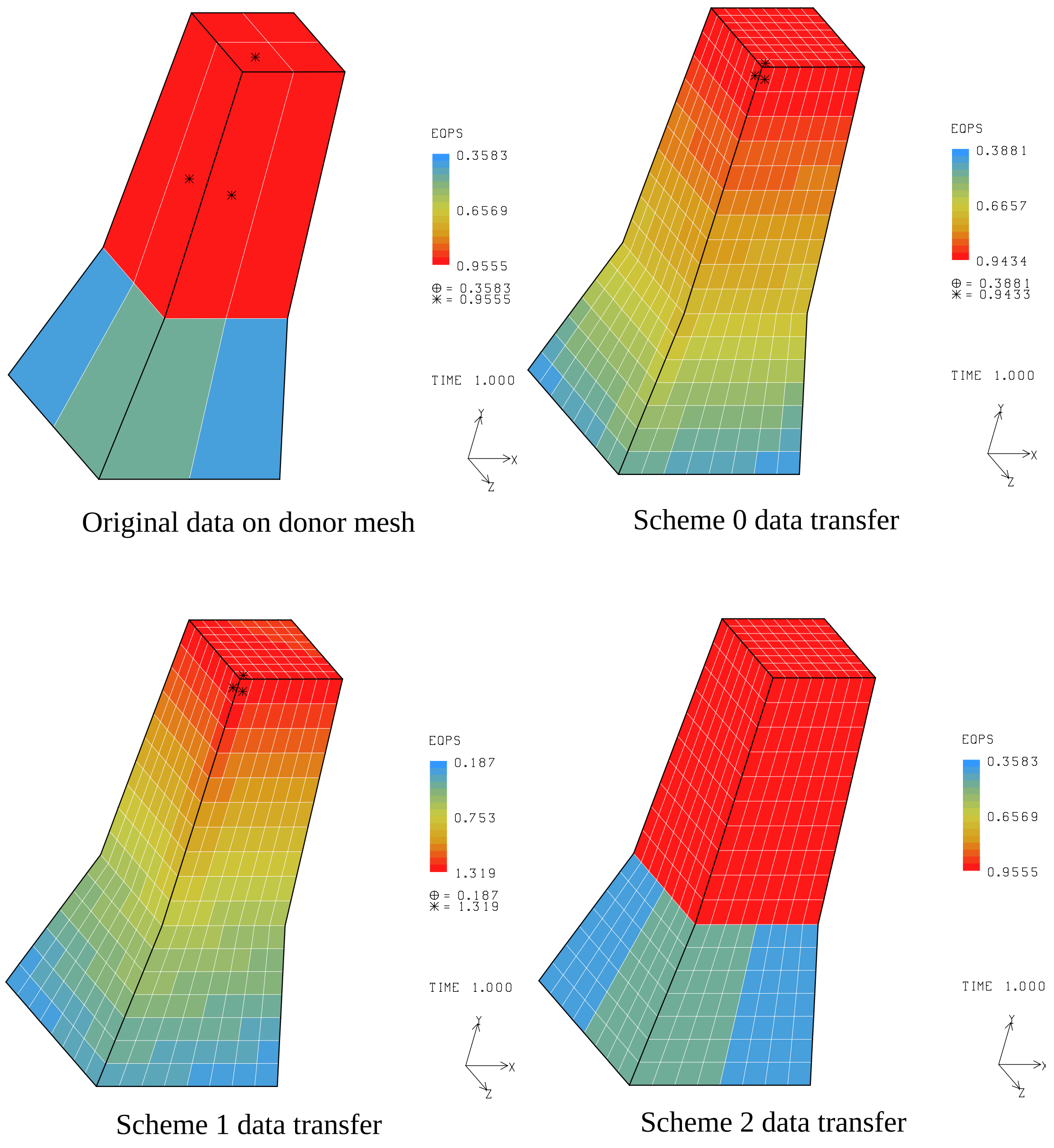

Figure 4. Data transfer comparison among interpolation schemes

As expected, the direct transfer option (lower right-hand corner) produces an identical plot, except for the mesh lines, to the original (upper left-hand corner). Schemes 0 and 1, on first glance appear to produce very similar results. However, the reader is advised to look closely at the legend on the color bands. Scheme 0 produces some dissipation of the original result (the minimum value of EQPS is greater than the original while the maximum value is less than the original). Scheme 1, on the other hand, extrapolates the result based on the element centroid to element centroid gradients in the original solution. Here the minimum value of EQPS is less than the original data while the maximum value of EQPS is much greater than the original data. The minimum and maximum values for the original data and each of the interpolation schemes are shown in Table 2. It should be kept in mind

Table 2. Comparison among interpolation schemes

	Minimum value of EQPS	Maximum value of EQPS
Original data	0.3583	0.9555
Scheme 0	0.3881	0.9434
Scheme 1	0.187	1.319
Scheme 2	0.3583	0.9555

that the colors shown in Figure 4 are based on the values of the variable at the element centroids (note the location of the stars on the figures). Therefore, it is reasonable to expect, that when additional sampling points are available, the interpolated data should exhibit a gradient similar to the original data. Thus, the author prefers the results obtained with scheme 1, explaining its selection as the default interpolation scheme.

This data transfer was accomplished with the following commands. First MAPVAR was executed (three times, once for each interpolation scheme):

```
mapvar test1
```

where test1.e is the donor mesh which contains the EQPS element variable results (shown in Figure 2) and test1.g is the recipient mesh (shown in Figure 3). At the command line prompt from MAPVAR, the commands,

```
sch 0
run
```

were entered for the first run. The default values for deformed processing, “def 1”, for the time step, last time step in file, for the searchbox, “sea 0.01”, and for the transfer map, “map 1 to 1”, were all accepted and thus no input was required for the commands activated by these keywords. In subsequent runs, the command “sch 0” was replaced by “sch 1” and “sch 2”. The results of the transfers (shown in Figure 4) were written to a file named test1.int. The text output file, test1.o, was also written but not used.

The second example problem demonstrates the use of transfer maps. Here the donor mesh consists of six element blocks (only one material) while the recipient mesh has only one element block. While this example problem may seem contrived, it is motivated by an actual application of data transfer between a radiation transport and a structural mechanics code. The actual application could not be shown here. Figure 5 shows the distribution of pressure over the six element blocks of the donor mesh. Figure 6 shows the interpolated distribution of pressure over the one element block of the recipient mesh. In Figures 5

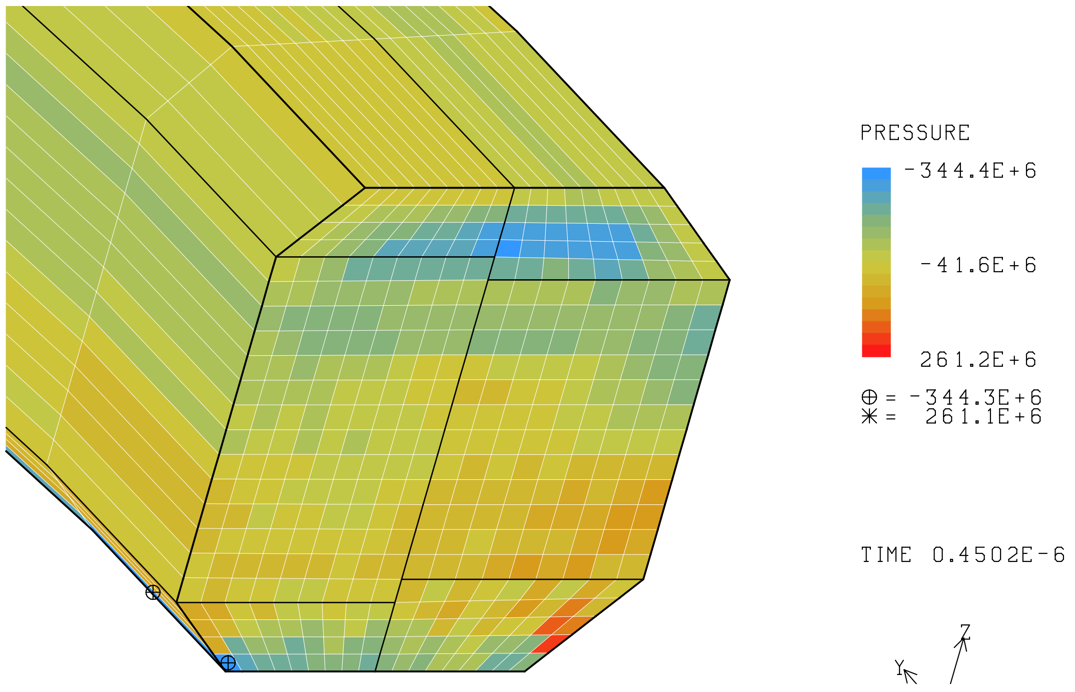

Figure 5. Donor mesh for transfer map demonstration

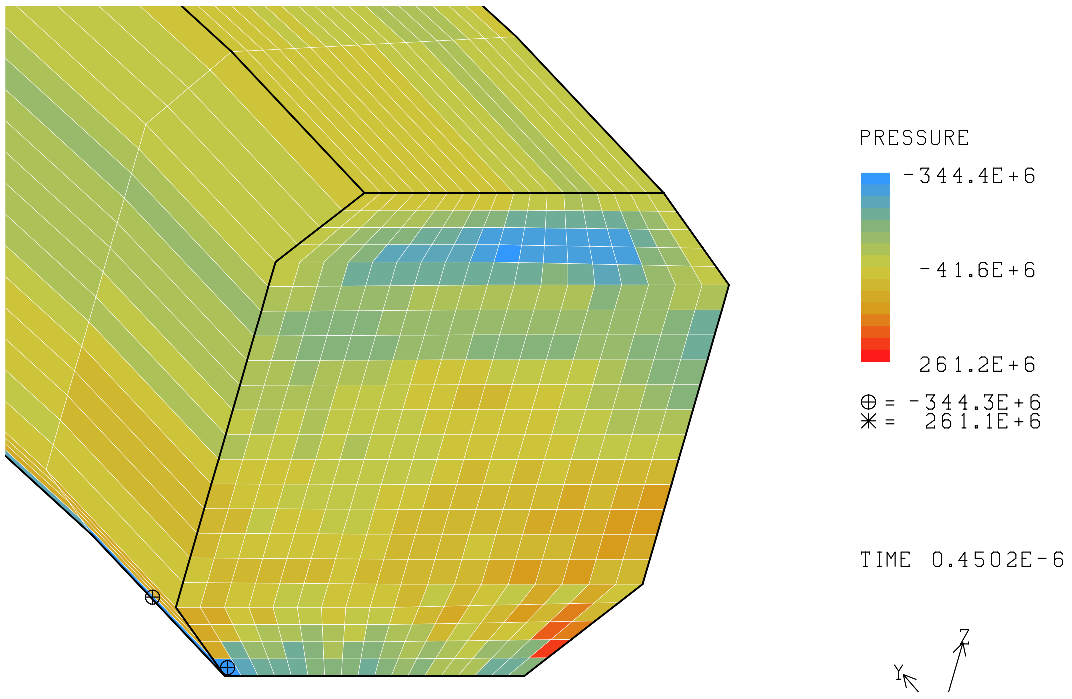

Figure 6. Interpolated results for transfer map demonstration

and 6, the white lines are the element boundaries while the black lines are the element block boundaries.

The MAPVAR commands that produced the map shown above were:

```
scheme 2
deform 0
times all
map 23 to 23
map 24 to 23
map 25 to 23
map 26 to 23
map 27 to 23
map 28 to 23
run
```

In this case, direct transfer was chosen because the donor and recipient meshes had virtually identical grid resolution. There were no displacements in the file so undeformed processing was appropriate. There was data available for three time steps on the donor mesh. The values shown came from the second time step. Selection of the second time step for display was arbitrary. The six element blocks of the donor mesh had element block identification numbers of 23, 24, 25, 26, 27, and 28. The one element block of the recipient mesh had an element block identification number of 23. Thus the transfer map provided for the transfer of data from element blocks 23 through 28 of the donor mesh into element block 23 of the recipient mesh. For this case, the command “map all to 23” would have accomplished the same effect.

References

[1]D. K. Gartling, “MERLIN - A Computer Program to Transfer Data Between Finite Element Meshes,” SAND81-0463, Sandia National Laboratories, Albuquerque, New Mexico, 1981.

[2]D. K. Gartling, “MERLIN - A Computer Program to Transfer Data Between Finite Element Meshes,” SAND89-2989, Sandia National Laboratories, Albuquerque, New Mexico, 1991.

[3]Schoof, L. A. and Yarberry, V. R., “EXODUS II: A Finite Element Data Model,” SAND92-2137, Sandia National Laboratories, Albuquerque, New Mexico, 1994.

[4]Mills-Curran, W. C, Gilkey, A. P., and Flanagan, D. P., “EXODUS: A Finite Element File Format for Pre- and Post-processing,” SAND87-2977, Sandia National Laboratories, Albuquerque, New Mexico, 1988.

[5]Flanagan, D. P., Mills-Curran, W. C., and Taylor, L. M., “SUPES A Software Utilities Package for the Engineering Sciences,” SAND86-0911, Sandia National Laboratories, Albuquerque, New Mexico, 1986.

[6]Sjaardema, G. D., “Overview of the Sandia National Laboratories Engineering Analysis Code Access System,” SAND92-2292, Sandia National Laboratories, Albuquerque, New Mexico, 1993.

[7]Swegle, J. W., Attaway, S. W., Heinstein, M. W., Mello, F. J., and Hicks, D. L., “An Analysis of Smoothed Particle Hydrodynamics,” SAND93-2513, Sandia National Laboratories, Albuquerque, New Mexico, 1994.

[8]Heinstein, M. W., Attaway, S. W. Swegle, J. W., and Mello, F. J., “A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes,” SAND92-2141, Sandia National Laboratories, Albuquerque, New Mexico, 1993.

[9]Schunk, P. R., Sackinger, P. A., Rao, R. R., Chen, K. S., and Cairncross, R. A., “GOMA - A Full-Newton Finite Element Program for Free and Moving Boundary Problems with Coupled Fluid/Solid Momentum, Energy, Mass, and Chemical Species Transport: User’s Guide,” SAND95-2937, Sandia National Laboratories, Albuquerque, New Mexico, 1995.

[10]Gilkey, A. P. “ALGEBRA - A Program That Algebraically Manipulates the Output of a Finite Element Analysis (EXODUS Version),” SAND88-1431, Sandia National Laboratories, Albuquerque, New Mexico, 1988.

[11]American National Standard Programming Language FORTRAN. American National Standards Institute, ANSI X3.9-1978, New York, 1978.

[12]Gilkey, A. P., and Glick, J. H., “BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis”, SAND88-1432, Sandia National Laboratories, Albuquerque, New Mexico, 1989.

APPENDIX A: Executing MAPVAR

Execution Script

MAPVAR is part of the SEACAS [6] system and is executed in a manner similar to the analysis programs in that system. A script is available to aid in the connection of files to the appropriate unit number. The script uses keyword options to define the particular file. The format of the script for MAPVAR follows:

```
mapvar [-file_options filename] [-options option] [--] [base]
```

Execution Files

MAPVAR requires two files, the files attached via the -plot and the -mesh file options. The file attached via the -plot option is the EXODUS file that has the nodal and element variable results that need to be interpolated onto the new mesh. This file is sometimes referred to as mesh-A or as the donor mesh. The file attached via the -mesh option is the GENESIS file of the new mesh. The nodal and variable results will be interpolated onto this file. This file is also referred to as mesh-B or the recipient mesh.

MAPVAR produces two files, the files attached via the -interpolated and the -output options. The file attached via the -interpolated option is the EXODUS file that contains the interpolated nodal and element variables from the -plot file on the mesh geometry from the -mesh file. This file is sometimes referred to as mesh-C. The file attached via the -interpolated option is the primary output from MAPVAR. The file attached via the -output option is a text file to which various run time information and error messages are written. A table of the file options used by MAPVAR is shown below.

Table A-1. File Options in MAPVAR

file_option	Argument	FORTRAN unit number	Default Extension	Type	Comments
-input	input_file	standard input	*base*.i	ASCII	Not used
-output	output_file	fort.7	*base*.o	ASCII	created
-plot	donor mesh	fort.12	*base*.e	EXODUS	required
-mesh	recipient mesh	fort.13	*base*.g	EXODUS	required
-interpolated	results mesh	fort.14	*base*.int	EXODUS	created

The MAPVAR script also has several command options. The -subroutine option will compile and link a subroutine in addition to the default executable or that replaces an already existing subroutine in the default executable. The primary use of this command option will be to replace the subroutines that apply values to nodes or elements not found in the search, see the section on User Subroutine for details. The -executable option is used to point to an alternate executable image of the program. The -help, -noexecute, and -aprepro command options will rarely if ever be utilized in connection with MAPVAR. A table of the command options used by MAPVAR follows:.

Table A-2. Command Options in MAPVAR

command_option	Argument	Default	Comments
-subroutine	subroutine_file	none	see User Subroutine
-executable	executable_image	ACCESS image	
-help	none	none	Not implemented
-noexecute	none	none	
-aprepro	aprepro_commands	none	Not used

Perhaps the simplest way to execute MAPVAR is by using the *base* concept. When executing MAPVAR in this manner only the *base* need be entered and the script will locate the proper files using default extensions. For example, the command line:

```
mapvar exam
```

is identical to the command line:

```
mapvar -plot=exam.e -mesh=exam.g -output=exam.o -interpolated=exam.int
```

Special Software

MAPVAR is written in ANSI FORTRAN-77 [11] with the exception of the following:

- File opening
- SUPES library [5] for dynamic memory allocation, and free field reader
- netcdf calls for EXODUSII [4]

APPENDIX B: Command Summary

MAPVAR was originally designed to be run interactively with command input from a terminal (standard-input). The input syntax is keyword command driven. Commands may be entered in any order. When all the input has been entered, the program may then be run or alternately terminated. In all cases, only the first three letters of any keyword are required. If the keyword requires a value in addition to the keyword itself, this is indicated by brackets. For example:

DEFormed <int>

The bold capital letters indicate the required portion of the keyword. The <int> indicates that the keyword **DEF**ormed command requires an integer value in the second field. A summary of the available commands follows:

HELp - provides a summary of commands to the terminal screen.

HELp <char> - provides a description of a specific command.

SCHeme <int> - defines the interpolation scheme to be used.

0 - simple interpolation

1 - linear least squares

2 - direct transfer

DEForm <int> - defines the coordinate space in which to work.

0 - original, undeformed geometry

1 - current, deformed geometry

2 - mesh annealing (deformed with displacements zeroed after map)

LISt **TIM**es - provides a list of time steps available from the donor mesh.

TIMe <real / **ALL**> - defines the time step to be interpolated or all time steps.

SEArchbox <real> - defines the search area around the donor mesh elements.

MAP <int or **ALL**> **TO** <int or **ALL**> **SCH**eme <int>- allows used to control the mapping from the donor mesh to the recipient mesh element block by element block. The **SCH**eme portion of this command is optional.

Defaults

SCHeme 1

DEForm 1

TIMes last time in the donor mesh data base

SEArchbox 0.01

MAP 1 **TO** 1; **MAP** 2 **TO** 2; **MAP** 3 **TO** 3; etc.

Constraints

SCHeme 2 is not available with 9-node quads.

Only **DEF**orm 0 is allowed with **TIM**es **ALL.**

If the **MAP** command is used, then the entire map must be input, the default setting is not updated but is instead erased.

This Page Intentionally Blank

Distribution:

Goodyear Tire and Rubber Co.
P.O. Box 3531
Akron Ohio 44309-3531

Attn:	Davis, Timothy
	Genbicki, Floyd
	Poldneff, Mike
	Benedict, Bob

Goodyear Technical Center
L-7750 Colmar-Berg
Luxembourg

Attn:	Jeusette, Jean-Pierre
	Quorin, Didier

	MS9042	08742	Chen, Er-Ping
	MS9042	08742	Jin, Paul S.
	MS9042	08742	Weingarten, Lawrence I.
	MS9405	08743	Neilan, Paul E.
	MS9042	08746	Kawahara, Wendell A.
	MS0841	09100	Hommert, Paul J.
	MS0834	09100	Baer, Mel R.
	MS0826	09100	Gartling, David K.
	MS0834	09100	Chu, T. Y.
	MS0828	09101	Bickel, Thomas C.
13	MS0826	09111	Hermina, Wahid and staff
27	MS0834	09112	Ratzel, Arthur C. and staff
19	MS0835	09113	Kempka, Steven N. and staff
	MS0827	09114	Griffith, Richard
	MS0825	09115	Rutledge, Walter H.
	MS0836	09116	Peterson, Carl W.
	MS0836	09116	Sundberg, David W.
32	MS0443	09117	Morgan, Harold S. and staff
25	MS0443	09117	Wellman, Gerald W.
	MS0557	09119	Baca, Thomas J.

12	MS0836	09121	Biffle, Johnny H. and staff
	MS0555	09133	Garrett, Mark S.
	MS1135	09134	Davis, David B.
	MS0553	09135	May, Rodney A.
	MS0553	09135	Jung, Joe
	MS0827	09136	Zepper, John D.
	MS0827	09136	Sjaardema, Greg
	MS0441	09226	Leland, Robert J.
	MS0819	09231	Peery, James S.
	MS0820	09232	Yarrington, Paul
19	MS0439	09234	Martinez, David R. and staff
	MS9018	8940-2	Central Technical Files
2	MS0899	4916	Technical Library
1	MS0619	15102	Review and Approval Desk For DOE/OSTI